Incremental update of electrostatic interactions in adaptively restrained particle simulations

Semeho Prince A. Edorh, Stephane Redon

Incremental Update of Electrostatic Interactions in Adaptively Restrained Particle Simulations

Semeho Edorh*, Stephane Redon †

April 9, 2018

Abstract

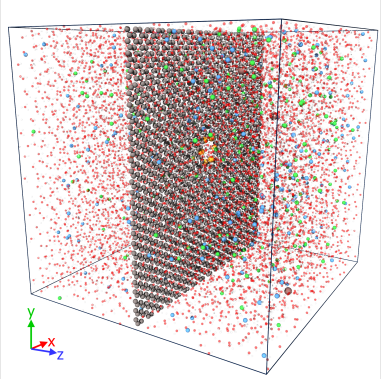
The computation of long-range potentials is one of the demanding tasks in Molecular Dynamics. During the last decades, an inventive panoply of methods was developed in order to reduce the CPU time of this task.

In this work, we propose a fast method dedicated to the computation of the electrostatic potential in adaptively restrained systems. We exploit the fact that, in such systems, only some particles are allowed to move at each timestep. We developed an incremental algorithm derived from a multigrid-based alternative to traditional Fourier-based methods. Our algorithm was implemented inside LAMMPS, a popular molecular dynamics simulation package.

We evaluated the method on different systems. We showed that the new algorithm's computational complexity scales with the number of active particles in the simulated system, and is able to outperform the well-established Particle Particle Particle Mesh (P3M) for adaptively restrained simulations.

Keywords: Adaptively Restrained Molecular Dynamics, Electrostatics, Multigrid, LAMMPS, Molecular Simulations

*Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
†Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France



Adaptively restrained simulation of a nanopore system which contains a carbon (gray) graphene sheet, water molecules (oxygen red, hydrogen white), sodium (cyan) and chlorine (green). Particles are driven through the pore by an external electric field. The flux of sodium is restricted by positively charged particles (orange) located at the edge of the nanopore. Counterions (brown) ensure the neutrality of the system. The simulation is sped up by adaptively restraining the motion of water molecules.

INTRODUCTION

Molecular Dynamics (MD) is the tool of choice in a large variety of domains such as biophysics, crystallography or material sciences[1,2]. MD simulations can be combined with (or replace) experiments for the study of large macro-molecular systems[3]. To simulate these systems, one may provide a description of the interactions that arise between the components of the studied system. This is done through interaction potentials. The evaluation of these potentials is often reported as the most computationally expensive part of an MD simulation[4]. Thus, reducing the cost of computing the potentials acting on different components of the system became a popular research topic. To overcome the quadratic scaling of long-ranged pairwise interactions such as electrostatics, sophisticated algorithms have been developed[5].

The $\mathcal{O}(N \log(N))$ mesh solvers in Fourier space such as Particle Particle Particle Mesh (P3M)[6,7] or Smooth Particle Mesh Ewald (SPME)[8], are the most popular approaches to the calculation of the electrostatic potential or forces in periodic systems. Although these Fast Fourier Fransform (FFT) based methods are often the less expensive solution for computing electrostatics on a single core, their performance on massively parallel architectures are limited by the all-to-all communications required by the FFTs[9]. This leads some authors to consider alternatives to these methods.

Hierarchical approaches like Fast Multipole Method (FMM)[10] are well known techniques for open systems. However these $\mathcal{O}(N)$ methods can be extended in periodic systems. FMM requires a very high order to obtain the smooth forces need for molecular dynamics. This increases its cost for MD simulations, especially with periodic boundary conditions. Other alternatives rely on iterative or multigrid-based methods[11–16].

Multigrid approaches utilize multiple grid levels with different spatial resolutions to compute the electrostatic interaction with a $\mathcal{O}(N)$ cost[17]. Despite their linear scaling, these mesh solvers in cartesian space are slower than the previously presented ones[5]. The Multilevel Summation Method (MSM) and the Meshed Continuum Method (MCM) are two of the most decent multigrid methods.

MSM directly calculates the potential on a grid by using several length scales. Calculations are spread over a hierarchy of grids, and the potential computed at coarse levels is successively corrected by contributions from finer levels up to the finest grid, which yields the final potential[13,18].

MCM uses densities with a compact support to sample the particles onto a grid and calculates an approximated potential by solving a Poisson equation in a multigrid fashion[14,19]. This potential is then interpolated at the particle positions. To retrieve the electrostatic potential for each particle, a short-range correction is computed based on neighboring particles. This step can take advantage of existing algorithms dedicated to short-range potentials, such as linked cells or Verlet lists.

Reducing the cost of evaluating the electrostatic potential at each time step is not the only way for accelerating particle simulations. Recently, an alternative solution, called adaptive restrained molecular dynamics (ARMD), was proposed in Artemova and Redon[20]. The method relies on a modification of the system's Hamiltonian to freeze and unfreeze particles temporarily during simulation based on their momenta, to avoid the need of updating interactions between frozen particles. When sufficiently many particles are restrained, the method may achieve significant speedups while still producing useful dynamics and statistics[20]. In order to take advantage of adaptive restraints, simulations must use incremental force update algorithms that skip the calculation of forces involving restrained particles. Although such algorithms have been demonstrated for short-range interactions[21], no method had yet been proposed for long-range interactions. In this paper, we propose a method to incrementally update electrostatic interactions in adaptively restrained simulations.

Some components of P3M rely on particle pair calculations which can be satisfactorily sped up with incremental computations. Conversely the expected speed up of the P3M Fourier part is limited. Among previous methods, MCM is a good candidate for incremental calculations since all of its CPU intensive components are linearly dependent on the number of particles. Therefore we were motivated to use this method which has a large room for improvement as regards of incremental calculations. We show that an incremental meshed continuum method (IMCM) can significantly speed up adaptively restrained simulations in

the presence of long-range interactions.

Below, we give a brief overview of adaptive restrained molecular dynamics (ARMD) as well as meshed continuum method (MCM). We then describe the incremental version of MCM. Finally, we compare this algorithm to the well known P3M on three various benchmarks.

METHODOLOGY

Adaptive Restrained Molecular Dynamics (ARMD)

In order to speed up calculations of the intermolecular potentials, ARMD was introduced [20 22]. In this model, particles with sufficiently large kinetic energy ($> \varepsilon^f$) are considered active, and have normal dynamics, while those with a kinetic energy below a given threshold ε^r are restrained, and stop moving completely.

Given a system of N particles, this behavior can be explained through Hamiltonian formalism[20], where the kinetic part of the Hamiltonian is modified, namely,

$$H_{AR}(\boldsymbol{q},\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^T\Phi(\boldsymbol{q},\boldsymbol{p})\boldsymbol{p} + V(\boldsymbol{q}) \tag{1}$$

where $\boldsymbol{q}$ is a $3N$-dimensional vector of coordinates, $\boldsymbol{p}$ is a $3N$-dimensional vector of momenta and $V(\boldsymbol{q})$ the interaction potential energy. The $3N \times 3N$ block-diagonal matrix $\Phi(\boldsymbol{q},\boldsymbol{p}) = diag[\Phi_1(\boldsymbol{q_1},\boldsymbol{p_1}),\ldots,\Phi(\boldsymbol{q_N},\boldsymbol{p_N})]$ represents the inverse inertia matrix which reads

$$\Phi_i(\boldsymbol{q_i},\boldsymbol{p_i}) = m_i^{-1}[1-\rho_i(K_i(p_i))]\boldsymbol{I}_{3\times 3} \tag{2}$$

where m_i, $\boldsymbol{q}_i$ and $\boldsymbol{p}_i$ are respectively the mass, position and momentum of particle i, $K_i(p_i)$ its kinetic energy and ρ_i is its restraining function. Given the thresholds ε_i^f and ε_i^r associated to the particle i, ρ_i is defined as follows:

$$\rho_i(k) = \begin{cases} 1 & \text{if } k \geq \varepsilon_i^f \\ 0 & \text{if } k \leq \varepsilon_i^r \\ s(k) & \text{if } \varepsilon_i^r \leq k \leq \varepsilon_i^f \end{cases} \tag{3}$$

where s is a $\mathcal{C}^2$ function that smoothly interpolates between 0 (*restrained dynamics*) and 1 (*full dynamics*).

One may incrementally update the interaction potential $V(\boldsymbol{q})$ since intermolecular potentials typically depend upon relative particle positions. Short-range particle-particle pair interactions such as Lennard-Jones potential were handled in [23]. By introducing a suitable neighbor list algorithm for active particles, they achieve a significant speed-ups, depending on the chosen thresholds.

We expect to broaden this result on systems where not only short-range interactions but also long-range electrostatics terms are needed.

Electrostatic potential

We want to compute the electrostatic potential Φ generated by N charged particles with charge q_i at position $\boldsymbol{x}_i$ in a cubic box $\mathcal{B}$ of length L. To ensure periodic boundary conditions (p.b.c), we assume that the simulation box $\mathcal{B}$ is replicated an infinite number of times [24] :

$$\Phi(\boldsymbol{x}) = -\frac{1}{4\pi\epsilon_0} \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \sum_{\substack{i=1 \\ \boldsymbol{x}_i^{\boldsymbol{0}}\neq\boldsymbol{x}}}^{N} q_i \frac{1}{||\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{n}}||}. \tag{4}$$

Here, $\boldsymbol{x}_i^{\boldsymbol{n}} = \boldsymbol{x}_i + \boldsymbol{n}L, \boldsymbol{n} \in \mathbb{Z}^3$ are the replicated particle images, and ϵ_0 is the dielectric permittivity. This potential can be seen as a solution of the following Poisson equation with respect to periodic boundary conditions

$$\Delta\Phi(\boldsymbol{x}) = -\frac{1}{\epsilon_0} \sum_{i=1}^{N} q_i \delta(||\boldsymbol{x} - \boldsymbol{x}_i||). \tag{5}$$

The right hand side (r.h.s) of this equation is described by Dirac δ functions associated to the point charges in the simulation box.

Given two successive timesteps ($t = t_0$) and ($t = t_1$), particles have two kinds of dynamics in ARMD. Whilst the so-called restrained charges are frozen between two successive timesteps, the active ones are allowed to move freely. Our goal is to take advantage of the fact that

some charges are restrained from t_0 to t_1 in order to incrementally update the potential Φ at t_1 knowing its value at t_0, thus speeding up the calculation of long-range forces.

In the upcoming section, we describe the main components of the meshed continuum method. Then, we propose an incremental version of this method.

The Meshed Continuum Method

The Meshed Continuum Method (MCM) was proposed by Bolten et al.[14,19]. The crux of the matter is to solve a modified version of the Poisson equation (5). The original expression of equation (5) cannot be solved numerically due to discontinuities introduced by Dirac δ functions[5]. Instead of modeling the particles by δ functions, one can introduce a smooth charge density ρ_{r_c} in the right-hand side (r.h.s) of the Poisson equation. The new distribution smears the charge of the particle over surrounding grid points. The induced potential can be computed numerically with a multigrid solver. This potential is interpolated at the particle's position. Finally a near-field correction is applied in the vicinity of each particle to minimize the error. This part can be handled explicitly via a neighbor list. In summary, MCM consists in five main phases :

- ▷ Evaluation of the **right-hand side** of the Poisson equation
- ▷ Computation of the **near-field correction** step
- ▷ Resolution of the Poisson equation via a **multigrid method**
- ▷ **Interpolation** of the solution at particles positions.
- ▷ **Computation of the potential and forces** by combining the near-field correction term and the interpolated solution.

The right-hand side — The smooth charge density ρ_{r_c} is defined as a normalized radially symmetric distribution such that $\rho_{r_c}(||\boldsymbol{r}||) = 0$ when $||\boldsymbol{r}|| > r_c$. The δ function associated to

each particle is replaced by $\varphi_i^{\boldsymbol{n}}(\boldsymbol{x}) = \rho_{r_c}(||\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{n}}||)$. Here, the normalization guarantees that the charge induced by $q_i\varphi_i^{\boldsymbol{n}}$ is equal to the charge q_i. Moreover, the compactness implies that the potential induced by the difference in charge distributions $q_i(\varphi_i^{\boldsymbol{n}} - \delta)$ is nil outside of the support of $\varphi_i^{\boldsymbol{n}}$[25]. Nonetheless, the optimal choice for the charge distribution is not clear. However, Bolten et al. proposed a centered quadratic B-spline as charge distribution. Moreover, a smooth charge density can be constructed with the help of cardinal B-splines. Precisely, a cardinal B-spline of order $m \in \mathbb{N}^*$ is a piecewise real function of class $\mathcal{C}^{m-2}$ with a limited support $[0, m]$. In practice, ρ_{r_c} can be constructed by centering, rescaling and normalizing a B-spline of a given order (See Appendix A). In addition, this B-spline can be described with m different polynomials of degree $m - 1$[26,27]. A practical algorithm for calculating the coefficients of polynomials, which determine a cardinal B-spline, was proposed by Milovanovic et al.[28]. In appendix A, we provide some examples of charge distributions.

Given a smooth charge density, the corresponding potential ϕ can be written as the convolution of ρ_{r_c} with the 3D Green's function :

$$\phi(\boldsymbol{x}) = \int_{\mathbb{R}^3} \frac{\rho_{r_c}(||\boldsymbol{y}||)}{4\pi||\boldsymbol{x} - \boldsymbol{y}||} \mathrm{d}\boldsymbol{y}. \tag{6}$$

When $r > r_c$ the induced potential is equal to $\frac{1}{4\pi||\boldsymbol{r}||}$. Depending on the charge distribution, this potential ϕ can be evaluated analytically.

By introducing the 3D Green's function in equation (4), the electrostatic potential can be split into a short-range and a long-range contribution[5]

$$\Phi(\boldsymbol{x}) = -\frac{1}{4\pi\epsilon_0} \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \sum_{\substack{i=1 \\ \boldsymbol{x}_i^{\boldsymbol{0}} \neq \boldsymbol{x}}}^{N} q_i \times \int_{\mathbb{R}^3} \left(\frac{\delta(\boldsymbol{y} - \boldsymbol{x}_i^{\boldsymbol{n}}) - \varphi_i^{\boldsymbol{n}}(\boldsymbol{y})}{||\boldsymbol{x} - \boldsymbol{y}||} + \frac{\varphi_i^{\boldsymbol{n}}(\boldsymbol{y})}{||\boldsymbol{x} - \boldsymbol{y}||} \right) \mathrm{d}\boldsymbol{y}. \tag{7}$$

The short-range term is given by

$$\Phi^{nf}(\boldsymbol{x}) := -\frac{1}{4\pi\epsilon_0} \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \sum_{\substack{i=1 \\ \boldsymbol{x}_i^{\boldsymbol{0}} \neq \boldsymbol{x}}}^{N} q_i \times \left(\frac{1}{||\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{n}}||} - \int_{\mathbb{R}^3} \frac{\varphi_i^{\boldsymbol{n}}(\boldsymbol{y})}{||\boldsymbol{x} - \boldsymbol{y}||} \mathrm{d}\boldsymbol{y} \right). \tag{8}$$

$$= \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \sum_{\substack{i=1 \\ \boldsymbol{x}_i^{\boldsymbol{0}} \neq \boldsymbol{x}}}^{N} C(\boldsymbol{x}_i^{\boldsymbol{n}}, \boldsymbol{x}) \tag{9}$$

where

$$\begin{aligned} C(\boldsymbol{x}, \boldsymbol{x}_i^{\boldsymbol{n}}) &= -\frac{q_i}{\epsilon_0} \times \left(\frac{1}{4\pi||\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{n}}||} - \int_{\mathbb{R}^3} \frac{\varphi_i^{\boldsymbol{n}}(\boldsymbol{y})}{4\pi||\boldsymbol{x} - \boldsymbol{y}||} \mathrm{d}\boldsymbol{y} \right) \\ &= -\frac{q_i}{\epsilon_0} \times \left(\frac{1}{4\pi||\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{n}}||} - \phi(||\boldsymbol{x} - \boldsymbol{x}_i^{n}||) \right) \end{aligned} \tag{10}$$

measures the error introduced while replacing a point charge located at $\boldsymbol{x}_i$ by a smooth density.

The near-field correction — As in P3M, the main advantage of introducing locally smeared charge distributions is the fact that they behave like Dirac densities beyond the cutoff distance: when $||\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{n}}|| > r_{max}$, $C(\boldsymbol{x}, \boldsymbol{x}_i^{\boldsymbol{n}}) = 0$. As result, the short-range contribution can be evaluated by taking into account only the interactions up to a given cutoff distance r_c[5,14]. Let us define $\mathcal{N}(\boldsymbol{x}) := \{\boldsymbol{x}_i^{\boldsymbol{n}}, \boldsymbol{n} \in \mathbb{Z}^3 / ||\boldsymbol{x} - \boldsymbol{x}_i^{\boldsymbol{n}}|| < r_c; \boldsymbol{x}_i^{\boldsymbol{n}} \neq \boldsymbol{x}\}$ as the set of all particles (including periodic images) in the neighborhood of $\boldsymbol{x}$. Therefore, using Equation (6), one can express this short-range contribution as follows:

$$\begin{aligned} \Phi^{nf}(\boldsymbol{x}) &= -\frac{1}{\epsilon_0} \sum_{\boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{x})} q_i \times \left(\frac{1}{4\pi||\boldsymbol{x} - \boldsymbol{x}_i||} - \phi(||\boldsymbol{x} - \boldsymbol{x}_i||) \right) \\ &= \sum_{\boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{x})} C(\boldsymbol{x}, \boldsymbol{x}_i) \end{aligned} \tag{11}$$

The computation of the short-range contribution can be handled efficiently with a neighbor list, which results in an optimal $\mathcal{O}(N)$ scaling.

Algorithm 1 Near-field correction $\Phi^{nf}(\boldsymbol{x}_i)$

1: **for** particle $\boldsymbol{j}/\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)$ **do**
2: $c \leftarrow -\dfrac{q_j}{4\pi\epsilon_0||\boldsymbol{x}_i - \boldsymbol{x}_j||} + \frac{q_j}{\epsilon_0}\phi(||\boldsymbol{x}_i - \boldsymbol{x}_j||)$ ▷ Compute the correction $C(i,j)$
3: $\Phi^{nf}(\boldsymbol{x}_i) \leftarrow \Phi^{nf}(\boldsymbol{x}_i) + c$
4: **end for**

The smooth component — The smooth long-range part in (7) corresponds to the solution of the modified version of equation (5) where Dirac impulses were replaced by a smooth

distribution.

$$\Delta\Phi^{sm}(\boldsymbol{x}) = -\frac{1}{\epsilon_0}\sum_{i=1}^{N} q_i\varphi_i^{\boldsymbol{n}}(\boldsymbol{x}) \tag{12}$$

The new formulation can be solved with any grid-based Poisson solver. This equation is discretized on a Cartesian grid Ω^h of spacing h. A fourth-order compact discretization of the equation (12) derived by Stephenson[29] leads to a linear system of equations, which can be solved using a multigrid algorithm. The partial differential equation (PDE) (12) can be solved in diverse ways. Nevertheless, due to its good scaling and its linear computational complexity, the multigrid approach is the method of choice for this work. This multi-resolution method speeds up the convergence of a basic iterative method (Gauss-Seidel, Jacobi, etc.) which efficiently reduces the high-frequency error. The multigrid method defines a hierarchy of coarser and coarser grids on which the low-frequency error at a given level is seen as a high-frequency one at a coarser level (see *e.g.* Trottenberg et al.[30] or Brandt et al.[31]).

The self-component — Sum (4) for the potential does not contain the self interaction $\boldsymbol{x} = \boldsymbol{x}_i$. In fact, the electrostatic potential due to a point charge diverges at the source's location. Conversely, a smooth charge distribution does not lead to a divergence of the potential. Moreover, the formulation (12) implies that Φ^{sm} sums the contributions of all charges including eventually $\boldsymbol{x} = \boldsymbol{x}_i$. Therefore, at each particle's location one must subtract a term corresponding to a self-interaction. This term, here denoted by Φ^{self}, is proportional to $\phi(\mathbf{0})$:

$$\Phi^{self}(\boldsymbol{x}_i) = \frac{q_i}{\epsilon_0}\phi(\mathbf{0}). \tag{13}$$

Equation (6) yields

$$\phi(\mathbf{0}) = \int_{\mathbb{R}^3} \frac{\rho_{r_c}(||\boldsymbol{y}||)}{4\pi||\boldsymbol{y}||}\mathrm{d}\boldsymbol{y}$$

Here, the integrand is a radial function. Therefore, a change of variables gives

$$\begin{aligned}\phi(\mathbf{0}) &= \int_{\mathbb{R}^+} r\rho_{r_c}(r)\mathrm{d}r \\ &= \int_0^{r_c} r\rho_{r_c}(r)\mathrm{d}r.\end{aligned} \tag{14}$$

Finally, the potential on a given particle i can be expressed as

$$\Phi(\boldsymbol{x}_i) = \Phi^{nf}(\boldsymbol{x}_i) + \Phi^{sm}(\boldsymbol{x}_i) - \Phi^{self}(\boldsymbol{x}_i). \tag{15}$$

Interpolation at particles positions — For a particle i, one can retrieve the smooth term $\Phi^{sm}(\boldsymbol{x}_i)$ by interpolating Φ_g^{sm} at the particle position $\boldsymbol{x}_i$, where Φ_g^{sm} is the solution of (12) on a grid

$$\Phi^{sm}(\boldsymbol{x}_i) \approx \tilde{\Phi}^{sm}(\boldsymbol{x}_i) = \sum_{\boldsymbol{m} \in \mathcal{I}(\boldsymbol{x}_i)} \omega_{\boldsymbol{m}}^{(i)} \Phi_g^{sm}(\boldsymbol{m}) \tag{16}$$

where $\mathcal{I}(x_i) \subset \Omega^h$ contains neighboring grid points of particle $\boldsymbol{x}_i$ and $\omega_{\boldsymbol{m}}^{(i)}$ are the associated weights. From the P closest grid nodes of particle i in each direction, a sub-mesh of P^3 neighboring points of i can be obtained. A similar stratagem was employed for P3M , where electrostatic forces were approximated with weights computed with splines[6]. In this work, $\Phi^{sm}(\boldsymbol{x}_i)$ was determined with a three dimensional polynomial interpolation[32,33]. The associated weights $\omega_{\boldsymbol{m}}^{(i)}$ depend on the atomic relative positions with respect of the grid. In addition, they are obtained by computing the tensor product of interpolation coefficients derived from the interpolation in each dimension.

The meshed continuum method can be summarized by the following algorithm:

Algorithm 2 Compute electrostatic potential $\Phi^(\boldsymbol{x}_i)$

$\rho^{sm} \leftarrow 0$ on the grid Ω^h
for each particle $\boldsymbol{i}$ **do**
    compute the near-field correction $\Phi^{nf}(\boldsymbol{x}_i)$ ▷ See algorithm 1
    $\Phi^{self}(\boldsymbol{x}_i) \leftarrow \frac{q_p i}{\epsilon_0}\phi(\mathbf{0})$ ▷ Self-correction
    Add $\boldsymbol{i}$'s contribution to smooth distribution ρ^{sm} on the grid Ω^h
end for
Solve $\Delta\Phi^{sm} = \rho^{sm}$ via a multigrid method
for each particle $\boldsymbol{i}$ **do**
    $\Phi^{sm}(\boldsymbol{x}_i) \leftarrow 0$
    for grid point $\boldsymbol{m} \in \mathcal{I}(\boldsymbol{x}_i)$ **do**
        $\Phi^{sm}(\boldsymbol{x}_i) \leftarrow \Phi^{sm}(\boldsymbol{x}_i) + \omega_{\boldsymbol{m}}^{(i)}\Phi^{sm}(\boldsymbol{m})$ ▷ Interpolate Φ^{sm}
    end for
    $\Phi(\boldsymbol{x}_i) \leftarrow \Phi^{nf}(\boldsymbol{x}_i) + \Phi^{sm}(\boldsymbol{x}_i) - \Phi^{self}(\boldsymbol{x}_i)$ ▷ Retrieve the electrostatic potential
end for

Incremental Meshed Continuum Method

Despite its linear scaling, the Meshed Continuum Method is slower than $\mathcal{O}(N\log(N))$ mesh solvers in Fourier space. Nevertheless, this $\mathcal{O}(N)$ complexity suggests that this method should be suitable for incremental update. The evaluation of the right-hand side of the equation (5) and the computation of the near-field correction are the most expensive components of the algorithm. In this section, we propose an incremental version of the MCM method which takes advantage of ARMD.

At $t = t_0$, we consider a system of N charged particles, among which R particles are restrained (frozen). Other particles are said to be active and able to move freely. Knowing the potential at $t = t_0$, we want to speed up the computation of the potential at $t = t_1$, knowing that the R restrained particles keep the same positions at $t = t_1$. The proposed algorithm scales linearly with the number of active particles $A = N - R$.

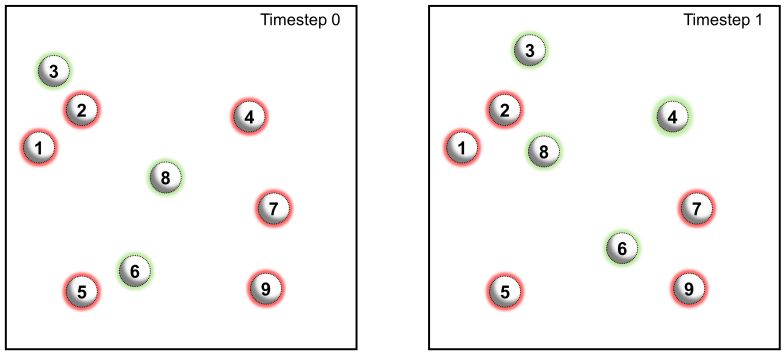


Figure 1: Two consecutive timesteps in ARMD. Left corresponds to ($t = t_0$) and right to ($t = t_1$). Red particles are restrained and cannot move. Green particles are active and are allowed to move freely.

Incremental computation of the right-hand side — The calculation of the right-hand side (r.h.s) of the equation (12) can easily be handled with a $\mathcal{O}(N)$ algorithm, since the number of grid points affected by each particle is smaller than the total number of grid points. The incremental calculation of the right-hand side is thus straightforward by splitting the potential into restrained and active contributions:

$$\rho^{sm} = \rho^{(Active)} + \rho^{(Restrained)} \tag{17}$$

At $t = t_0$ the term $\rho^{(Restrained)}$, which corresponds to the r.h.s associated to restrained particles, is stored, and does not need to be updated at $t = t_1$. Only $\rho^{(Active)}$ needs to be computed at each timestep. Assuming the overhead resulting from (17) is small, the calculation has an $\mathcal{O}(A)$ complexity. In practice, two separate grids ($\mathcal{G}_{all}$ and $\mathcal{G}_{restrained}$) have to be used. $\mathcal{G}_{restrained}$ holds the contribution $\rho^{(Restrained)}$ obtained by running over all restrained particles in the system and sum successively charge contributions on the grid according to their relative location on the mesh. $\mathcal{G}_{all}$ corresponds to the contribution of all particles ρ^{sm}. At the initial timestep, $\mathcal{G}_{restrained}$ is computed and saved in memory. For each timestep, one can copy values from $\mathcal{G}_{restrained}$ into $\mathcal{G}_{all}$. Then, the contribution of all active

particles is added to $\mathcal{G}_{all}$ in order to obtain ρ^{sm}. Consequently, this strategy may significantly reduce the CPU cost of the evaluation of ρ^{sm} by doubling the required memory.

In an adaptively restrained simulation, the distribution of active and restrained particles is not constant. At each time step, some restrained particles may become active, and vice versa (*e.g.* Particle 4 in Figure 1). At the next timestep $t = t_2$, the restrained component thus needs to be updated. This leads to a $\mathcal{O}(S)$ task, where S is the number of switching particles. Typically, $S << A < N$, and the extra cost can be neglected.

Incremental Near-field correction — Singh et al. proposed an incremental algorithm to tackle the computation of short-range pairwise potential in ARMD[23]. The near-field correction presented in the algorithm 1 can be seen as a short-range pairwise potential. In fact, from (10), the total of near-field corrections needed to be applied to a given particle i can be obtained by the sum $C(i,:) = \sum_j C(i,j)$, where i and j are neighbors.

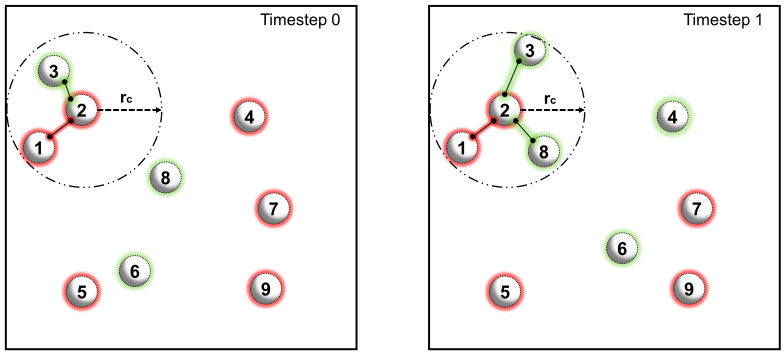


Figure 2: Near correction for particle (2) at ($t = t_0$) (left) and ($t = t_1$) (right). **Restrained-Restrained** corrections (red links) are unmodified between two successive time steps. **Active-Restrained** corrections (green links) have to be updated.

In the considered system, two kinds of correction have to be taken into account (Figure 2):

▷ The correction where both particles are *restrained*: **Restrained-Restrained** correc-

tions

- ▷ The correction where at least one particle is *active*: **Active-Restrained** or **Active-Active** corrections

Restrained-Restrained corrections can be stored between two consecutive timesteps, since their value does not change. Conversely, other corrections need to be recomputed. Therefore, one can split $C(i,:)$ as a sum of **Restrained-Restrained** corrections on particle i ($C(i,\mathcal{R})$) and **Active-Restrained** or **Active-Active** corrections on particle i ($C(i,\mathcal{A})$):

$$\begin{aligned} C(i,:) &= C(i,\mathcal{R}) + C(i,\mathcal{A}) \\ &= \sum_{j\in\mathcal{N}(i)} C(i,j)\mathbb{1}^{\mathcal{R}}(i)\mathbb{1}^{\mathcal{R}}(j) + \sum_{j} C(i,j)(1-\mathbb{1}^{\mathcal{R}}(i)\mathbb{1}^{\mathcal{R}}(j)). \end{aligned} \tag{18}$$

Here $\mathbb{1}^{\mathcal{R}}$ corresponds to the characteristic function of the set of restrained particles.

While $C(i,\mathcal{R})$ remains unchanged between $t=t_0$ and $t=t_1$, $C(i,\mathcal{A})$ need to be computed. When the particle i is restrained, $C(i,\mathcal{R})$ corresponds to the sum of corrections that involve only restrained neighbors of i and $C(i,\mathcal{A})$ sums the contributions from its active neighbors. Obviously, when i is active, $C(i,\mathcal{R})$ is nil and $C(i,\mathcal{A})$ is obtained by summing the contributions from all neighbors of i.

The superfluous computation of interactions between restrained particles can be avoided by storing for each particle the **Restrained-Restrained** component of the correction (See Algorithm 3). For a restrained particle, the missing contribution can be computed at each timestep by looping only on its *active* neighbors. Furthermore, the incremental near-field correction can be done more efficiently by modifying the neighbor list of each particle such that it carries only the information on the needed pairwise corrections[34].

Algorithm 3 Compute the **Restrained-Restrained** component $\Phi^{(nf,R)}(\boldsymbol{x}_i)$

if $\boldsymbol{i}$ is *restrained* **then**
  for **restrained** particle $\boldsymbol{j}/\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)$ **do**
    $\Phi^{(nf,R)}(\boldsymbol{x}_i) \leftarrow \Phi^{(nf,R)}(\boldsymbol{x}_i) - \dfrac{q_j}{4\pi\epsilon_0||\boldsymbol{x}_i - \boldsymbol{x}_j||} + \frac{q_j}{\epsilon_0}\phi(||\boldsymbol{x}_i - \boldsymbol{x}_j||)$
  end for
else
  $\Phi^{(nf,R)}(\boldsymbol{x}_i) \leftarrow 0$
end if

As mentioned above, usually there is also a small number of particles which switch state between consecutive timesteps. A special treatment needs to be done for these particles. At each timestep, the correction between switched particles and restrained particles must be removed from or added to the **Restrained-Restrained** component depending on the nature of the switch (« *Restrained* to *Active* » or « *Active* to *Restrained* »). Therefore the incremental near correction is an $\mathcal{O}(A + S)$ task.

Algorithm 4 Update for switched particles

for each *switched* particle $\boldsymbol{s}$ **do**
  if $\boldsymbol{s}$ is *restrained* –> *active* **then**
    Remove contribution to $\rho^{(sm,R)}$
    $\Phi^{(sr,R)}(\boldsymbol{x}_s) = 0$
  else
    Add contribution to $\rho^{(sm,R)}$
    Compute $\Phi^{(nf,R)}(\boldsymbol{x}_s)$ ▷ See Algorithm 3
  end if
end for

The interpolation coefficients — A final word is on the computation of the interpolation coefficients $(\omega^{(i)}_{\boldsymbol{m}})_{\boldsymbol{m}\in\mathcal{I}(\boldsymbol{x}_i)}$. The computation of these weights can be done on-the-fly as in the original Meshed Continuum Method. However, we chose to precompute them at the first timestep. In practice, since these weights can be obtained through tensor product, only

mono-dimensional coefficients are stored. The memory requirement of this approach is $3P$ coefficients per particle (instead of P^3). A slight overhead is therefore generated, but this approach might be useful on large systems. In addition, saving interpolation coefficients is a suitable option for adaptive restrained molecular dynamics. In fact, these weights can be reused as long as the corresponding particle does not move. As a consequence, only coefficients associated to *active* particles require a frequent update.

Withal, the Incremental Meshed Continuum Method is summarized with the algorithm 5.

Algorithm 5 Incremental electrostatic potential $\Phi(\boldsymbol{x}_i)$

Require: $(\omega_{\boldsymbol{m}}^{(i)})_{\boldsymbol{m}\in\mathcal{I}(\boldsymbol{x}_i)}$, $\Phi^{(nf,R)}(\boldsymbol{x}_i)$, $\rho^{(sm,R)}$, $\Phi^{self}(\boldsymbol{x}_i)$.

1: **for each** *restrained* particle $\boldsymbol{r}$ **do**
2: $\quad\Phi^{nf}(\boldsymbol{x}_r) = \Phi^{(nf,R)}(\boldsymbol{x}_r)$
3: $\quad$Update near-field correction for only active neighbors $\Phi^{nf}(\boldsymbol{x}_r)$
4: **end for**
5: $\rho^{sm} \leftarrow \rho^{(sm,R)}$ on the grid Ω^h
6: **for each** *active* particle $\boldsymbol{a}$ **do**
7: $\quad$Add contribution to smooth distribution ρ^{sm} on the grid Ω^h
8: $\quad$Update interpolation weights $(\omega_{\boldsymbol{m}}^{(a)})_{\boldsymbol{m}\in\mathcal{I}(\boldsymbol{x}_a)}$
9: $\quad$Update near-field correction $\Phi^{nf}(\boldsymbol{x}_a)$ ▷ See Algorithm 1
10: $\quad\Phi^{self}(\boldsymbol{x}_a) \leftarrow q_a\phi(0)$ ▷ Update self-correction
11: **end for**
12: Solve $\Delta\Phi^{sm} = \rho^{sm}$ via multigrid method
13: Add all contributions to $\Phi(\boldsymbol{x}_i)$ ▷ Analogous to lines [8-14] of Algorithm 2
14: Update *active* and *restrained* lists
15: Update for switched particles ▷ See Algorithm 4

Parallelization

We implemented the IMCM in the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS)[35,36], a well-established simulation package. For completeness, this section

presents a brief overview of the parallelization capabilities of the presented methods.

The meshed continuum method can be parallelized efficiently with the help of a standard domain-decomposition scheme where the physical domain, partitioned as sub-domains, is distributed onto the available processors. LAMMPS exploits the Message Passing Interface (MPI) standard and the Recursive Coordinate Bisection (RCB) algorithm to distribute sub-domains (i.e. subsets of particles and grid points) over MPI processes. Usually, all computations required by a geometric multigrid method can be described with the help of stencils operators, which often have a compact support[37]. Thus, each process can apply these stencils on grid points located inside its sub-domain. However, for a given process, the computations on grid points that are situated close to the sub-domain boundary may require grid points that are held by nearby sub-domains. Therefore, to ensure that the necessary data is available during computations, each sub-domain is enlarged by a surrounding *ghost* area.

To evaluate the right-hand side of Equation (12), each process computes the needed contribution on the grid points it owns (including ghost ones). Then, a *backward* communication step is applied. In order to fully sum contributions in their domains, all processes communicate the quantities of their ghost grid points and accumulate them into their own *real* grid points. Periodic boundary conditions are taken into account if needed.

Near-field contributions obey the same principles. They share the same computational pattern as short-range potentials in LAMMPS. Each process computes interactions between its particles (both non-ghost and ghost) through neighbor lists. Then, ghost particles contributions are communicated to the corresponding non-ghost particles.

The implemented multigrid solver follows the popular V-Cycle strategy[17,30]. In order to travel through the hierarchy of grids defined by the multigrid method, restriction and prolongation operations must be applied. These operations can be formulated as stencil operators and can be locally applied on each sub-domains. Then, a global communication is required after each restriction or prolongation operation[14,38]. In addition, the discretized Poisson problem can be described, on each grid level, by a stencil operation. Therefore,

additional communications are required. Although this parallelization scheme can be handle with ease, the multigrid method requires more attention, especially on large clusters. In fact, as the problem size is reduced on coarser grids levels, it may arise that the number of unknowns exceed the number of available process. Various strategies have been proposed in order to tackle this issue[38]. We choose to let the processes without assigned unknowns stay idle during the computation on these coarser levels. It should be noted that this approach introduces load-imbalance.

Finally, each process retrieves forces by deriving the obtained solution at its own grid points. A *forward* communication step is performed in order to fill the ghost grid cells surrounding each sub-domain.

Parallelization capabilities presented above can be effortlessly extended to IMCM. For instance, incremental computation of the near-field contributions can be done as follows: each MPI process computes corrections acting on its own particles (including ghosts), while avoiding computations between restrained particles. This is similar to what is proposed in[39] to parallelize the calculation of short-range interactions in ARMD.

Benchmarks

In order to evaluate the method, we designed three test systems containing sodium (Na) and Chlorine (Cl) particles:

- System A (64000 particles) - corresponds to a solid sodium chloride. In a $112.8 \times 112.8 \times 112.8$ Å^3 periodic box, 32000 NaCl pairs were placed according to the halite or rock-salt crystal structure. We evaluated our implementation (accuracy and CPU time) of the meshed continuum method by running NVE molecular dynamics on system. Then, the performance of the incremental meshed continuum method was assessed. To do so, we ran several simulations where, between two consecutive timesteps, a percentage (ranged from 0 to 100%) of randomly selected particles was restrained. Thus, by mimicking the behavior of ARMD, we could easily understand the acceleration of various component

of the IMCM. This system was also used to evaluate the parallelization of IMCM. Precisely, we replicate the system twice in each direction (512000 atoms).

- System B (23232 particles) - corresponds to a sodium chloride in the aqueous solution at 10.0 molality concentration. 6912 water molecules were combined with 1248 NaCl pairs in a $65.26 \times 65.26 \times 65.26$ Å^3 periodic box. The system B was maintained at 298K with a Langevin thermostat[22] and thermodynamic properties were computed. During these simulations, water particles dynamics were adaptively controlled via various energy thresholds $(\varepsilon^r, \varepsilon^f)$. Meanwhile Na and Cl particles are always active — $(\varepsilon^r, \varepsilon^f)_{Na;Cl} = (0; 0)$. $(\varepsilon^r, \varepsilon^f)$ affects the average number of restrained particles. Therefore, one may carefully choose them. Thus, we evaluated the ability of the IMCM to speed up adaptively restrained simulations of coulombic systems.

- System C (26290 particles) - Here we studied the interaction of NaCl and water ions with a monolayer porous graphene in cubic box with a $L = 70$ Å edge . A graphene layer with a 7 Å -diameter nanopore is placed at the center of the system in the plane $(z = 0)$. An external electric field of $1\,\mathrm{V/Å}$ applied along z-direction drives particles through the nanopore. 6 charged particles are placed at the edge of the pore. Particles in the graphene sheet, except those forming the pore, are charge free (Fig. 3). In order to realize the neutrality, 4 extra counter-ions were added in the system. Furthermore, the system contains also 8000 water molecules, 250 sodium chloride pairs and 1780 carbon atoms.

  During the simulation, the carbon atoms of graphene layer were frozen without thermal vibration. It was shown that this has a minor impact on the overall dynamics[40,41]. Similarly to system B, sodium chloride ions are always active whilst water molecules follow adaptively restrained molecular dynamics. Transport properties of this system were analyzed for different combinations of energy thresholds $(\varepsilon^r, \varepsilon^f)$. We expected to observe a so-called concentration polarization layer (CPL) in the vicinity of the graphene layer[40] as well as the ion selectivity of the pore[42,43].

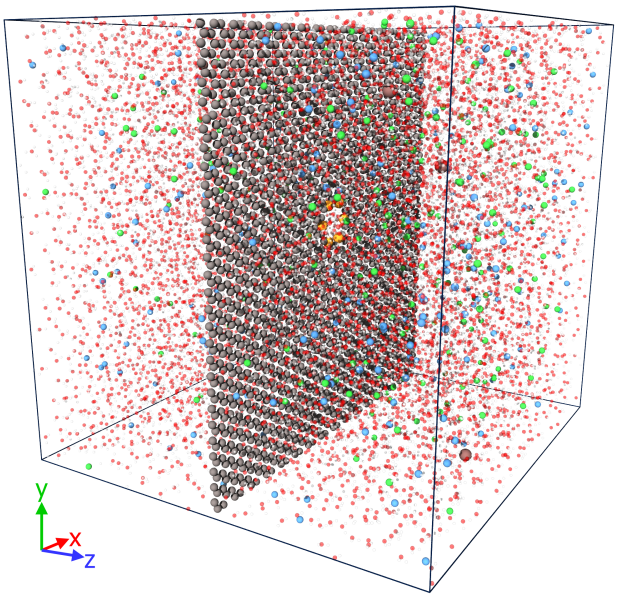


Figure 3: Sketch of the nanopore system. Gray color stands for carbon particles of the graphene sheet. Red and white stand for oxygen and hydrogen respectively. Sodium atoms are represented in cyan color while chlorine particles are green. Particles are driven through the pore by an external field applied along z-direction. The flux of sodium atoms is restricted by positively charged particles (orange) located at the edge of the nanopore. Counterions (brown) ensure the neutrality of the system.

Interaction Potentials — Since previous quoted systems were related to sodium chloride, we used the NaCl/ϵ force-field proposed by Fuentes-Azcatl and Barbosa[44] to describe intermolecular interactions . The advantages of this non-polarizable force-field are its simplicity and its ability to reproduce experimental results. The employed force-field is based on a set of radial particle-particle pair potentials involving Lennard-Jones (LJ) and electrostatic contributions. They used screening factors in the electrostatic interactions to account for the effect of polarization and they showed that in conjunction with the TIP4P/ϵ or the SPC/ϵ water force-fields, their model had a good agreement with experimental data at 298.15 K[44].

Armed with this conclusion, we described intermolecular interactions between particles included Lennard-Jones and electrostatic potentials with scaled charges, as shown in equation (19).

$$U_{ij} = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}\lambda_i\lambda_j \tag{19}$$

where q_i represents the electric charge of particle i, r_{ij} is the distance between particle i and particle j , σ_{ij} represents the LJ separation distance and ϵ_{ij} is the depth of the LJ potential well. λ_i is the scaling factor introduced to account for the effect of polarization. The parameters used in this research are summarized in table 1.

atom	q_i (e)	λ_i	σ_{ii} (Å)	ϵ_{ii} (kcal/mol)
Na	1.000	0.885	2.520	0.003
Cl	-1.000	0.885	3.850	0.382
H	0.445	1.000	0.000	0.000
O	-0.890	1.000	3.188	0.169
C	0.000	1.000	3.550	0.074
$^{*}\mathbf{C}^{(a)}$	1.5	1.000	3.550	0.074
$\mathbf{X}^{(b)}$	-2.25	1.000	3.550	0.074

[a] C_* corresponds to charged particle forming the nanopore; [b] X are counter ions.

Table 1: Values of potential parameters: C corresponds to charged particle forming the nanopore and X are counter ions.

Assuming that the pure water and the ions potentials are compatible, one may use the Lorentz-Berthelot mixing rules for calculating ϵ_{ij} and σ_{ij} (Equation (20)).

$$\epsilon_{ij} = \sqrt{\epsilon_{ii}\epsilon_{jj}} \quad ; \quad \sigma_{ij} = \left(\frac{\sigma_{ii} + \sigma_{jj}}{2}\right) \tag{20}$$

It is straightforward that the treatment of the coulombic term in (19) is similar to the work previously presented (*cf.* Algorithm 5). Thus, our method can be employed in order to compute incrementally electrostatic component of (19).

Finally, water molecules were treated as rigid body using SPC/ϵ model and harmonic potentials were used for bonds and angles (See table 2)[44].

Potentials	Equilibrium value	Prefactor
O-H Bond	0.957 Å	450 kcal /mol/Å^2
H-O-H Angle	109.47°	55 kcal /mol/$^{\circ 2}$

Table 2: Parameters of harmonic bond and angle potentials in SPC/ϵ water

Implementation details — The Incremental Meshed Continuum method was implemented in the 1^{st} Feb 2014 LAMMPS[45]. Most simulations were performed on a single core on a Dell Precision M4700 laptop with an Intel® Core ™ i7-3840QM CPU @ 2.80GHz. Parallel Benchmarks were performed using a cluster with 8 nodes equipped with 8/16 CPUs Intel Xeon E5540 and a Gigabit Ethernet network. We run the benchmark both for IMCM and for P3M on one node with different number of processes.

We tested various orders of B-spline distributions for the computation of the right hand side. An alternative to these piecewise functions consists in using high-order polynomial functions on $[0, r_c]$. In most cases, the 5^{th}-order B-spline or the 10^{th}-order polynomial function were the best choices. To speed up these functions calls, we used a look-up table with respect to the squared distance. The use of a look-up table reduces by more than a factor 3 the cost of the right-hand side's computation. We computed the near-field correction (11) based on a neighbor-list algorithm implemented within LAMMPS[36]. A similar treatment was done for the Lennard-Jones terms in equation (19).

In MD, forces must be computed along with the potential: $F_i = -q_i \nabla \Phi(\boldsymbol{x_i})$. As for particle mesh methods, there is no unique way to achieve this derivation[9]. Analytic differentiation of interpolation weights can be used for this purpose. However, this approach conserves energy but not momentum. One can reduce the momentum drift by removing the mean force. This yields to the non-conservation of energy. A second alternative is to directly derive Φ^{sm} with the use of finite difference operators[46]. Then, the polynomial interpolation scheme

is applied to evaluate smooth forces at each particle's position. This approach conserves the momentum while breaking the energy conservation. Fortunately, this drawback can be reduced by applying a mass-weighted correction[47].

In order to validate our implementation, we used P3M to compute reference forces with a 10^{-10} relative error. Then we evaluated various cutoff radii r_c, grid sizes, interpolation orders, and charge densities (Figure 5) to find the optimal configuration for MCM. Finally, we compared the associated CPU processing time to that from P3M solver with tantamount accuracy.

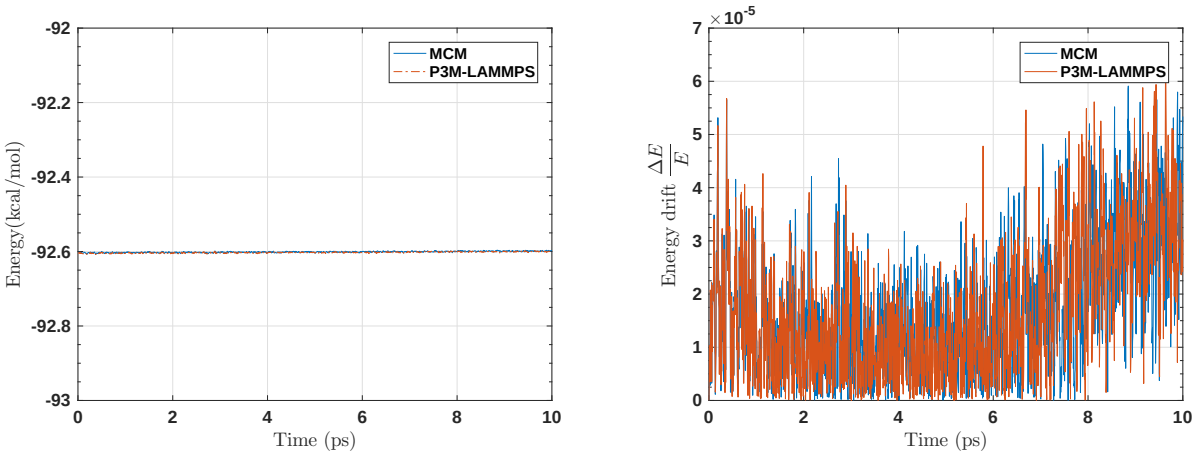
RESULTS

Solid Sodium Chloride Crystal

MCM was tested with various charge densities (B-splines or polynomials) on the system A using standard molecular dynamics. For this pure NaCl system of 64000 particles, the choice of the 10^{th}-order polynomial as charge density seems to be the best option for MCM (Figure 5a). Several grid sizes were also evaluated. This leads us to the conclusion that for the system A, the 64^3 grid is the fastest choice for low-accuracy simulations while a 128^3 grid is efficient for high-quality ones. Roughly, the use of a lower grid size will require a larger cutoff radius. Therefore, the processing time of the method will be dominated by the computation of the near-field correction. A bigger grid size allows the use of a smaller cutoff. The cost of the near-field correction is indeed reduced but the overall cost is governed by the calculation of the right-hand side (Figure 6) which can be pricey.

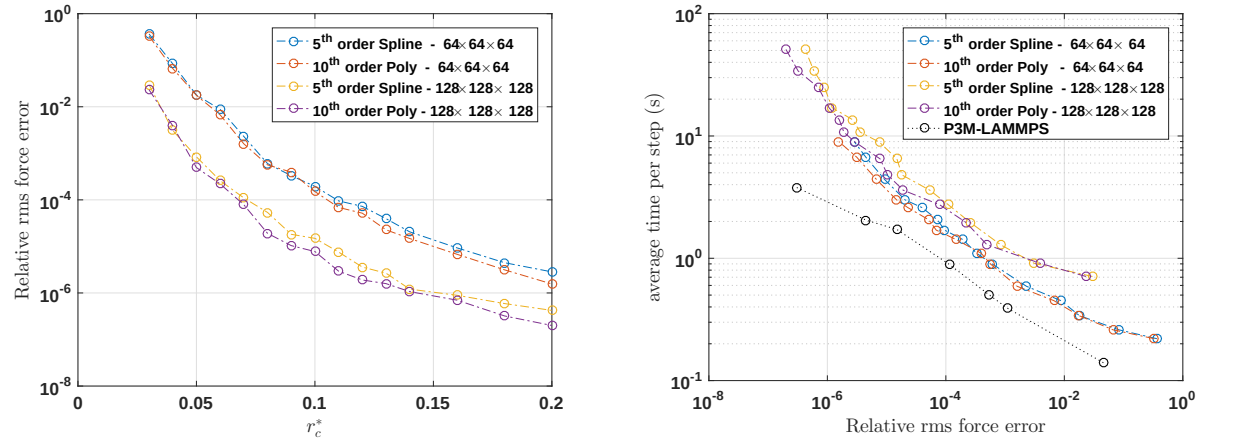
NVE simulations were conducted with the meshed continuum method (MCM) in order to confirm the quality of our implementation. For the same order of accuracy, we had similar energy conservation for both P3M and MCM (Figure 4). However, for a given accuracy MCM is 2 to 4 times slower than P3M (Figure 5b). These results are slightly better in comparison to the previous state-of-the-art benchmarks[5]. For the purpose of a fair comparison, P3M

was tuned in order to achieve the best performance for the given error criterion.



(a) Energy. (b) Energy drift.

Figure 4: The evolution of the energy E per particle over 10ps for a NVE simulation of 64000 particles with time-step increment $\Delta t = 2$fs. MCM is compared to P3M at similar accuracy ($\sim 10^{-5}$). The energy drift $|\Delta E/E|$ of both methods is also plotted (right).



(a) Relative RMS force error versus the cutoff radius. (b) Required CPU time versus the relative RMS error in forces.

Figure 5: Evaluation of our implementation of MCM in LAMMPS on System A (64000 particles). The 10^{th} order polynomial function and the 5^{th} order B-spline were compared on 64^3 and 128^3 Grid. $r_c^* = \dfrac{r_c}{L}$ is the normalized radius of the charge density.

The incremental version of the meshed continuum method was applied to several configura-

tions of the system A, where some particles are restrained between two consecutive timesteps. Various percentages ranging from 0 to 100% of active particles were tested in order to validate the method. Here, particles, under restraints, were constraints at their location without ARMD. This allows us to explore, with ease, diverse distribution of active particles.

As expected, the IMCM scales linearly with the number of active particles. Furthermore, when a sufficient number of particles are restrained, the IMCM algorithm is able to outperform the standard P3M algorithm (precisely, when less than 30% of particles are active in this benchmark (Figure 6). For a configuration where less than a tenth of particles are active, one can expect at least a $\times 2$ speedup relatively to the Particle Particle Particle Mesh (P3M).

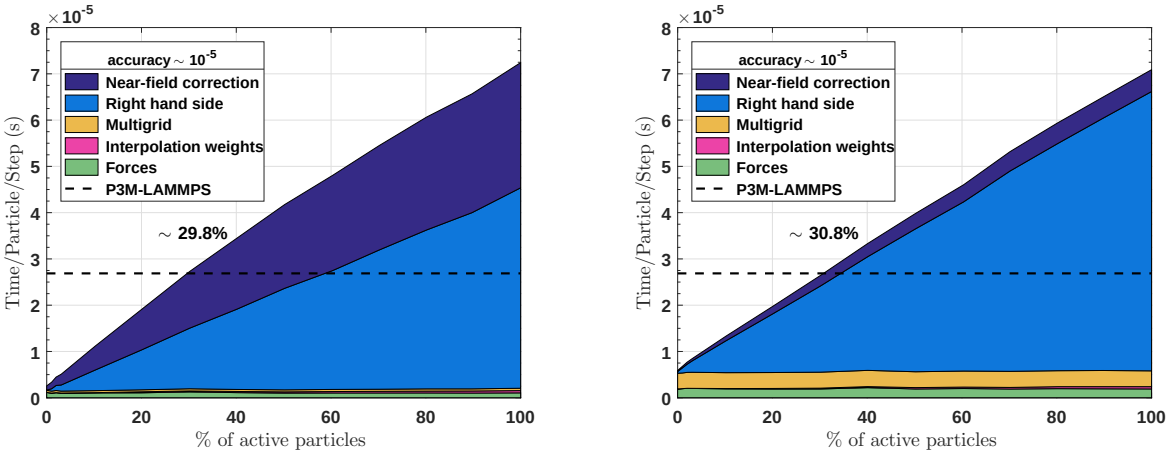
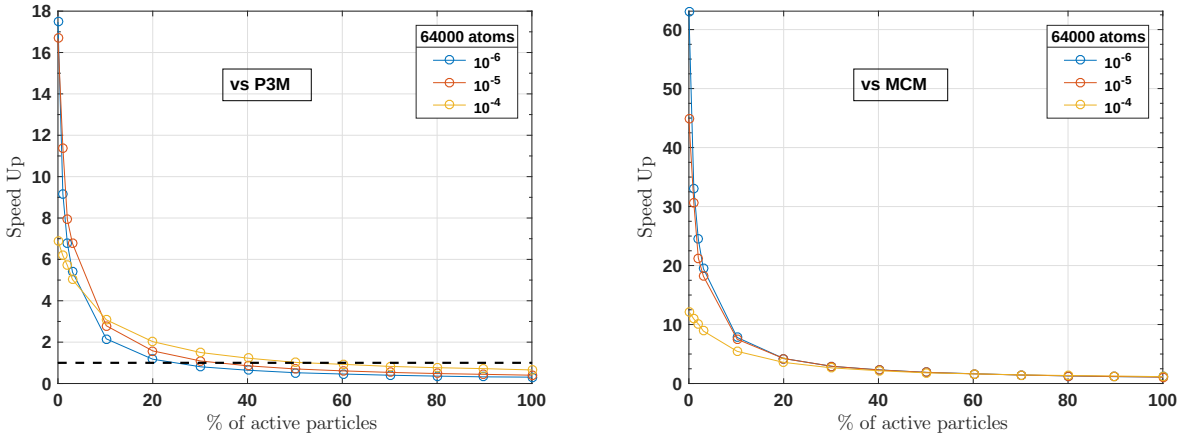


Figure 6: System A - Runtime of Incremental Meshed Continuum. Left : 64^3 grid and normalized cutoff $r_c^* = 0.16$. For each particle, the evaluation of the smooth density requires $\sim 20^3$ grid points. Right: 128^3 grid and $r_c^* = 0.09$ ($\sim 23^3$ grid points per particle).

Figure 7 confirms that the IMCM is a valuable alternative to P3M methods when an update of forces is needed. When the system is simulated at a relatively good accuracy ($\sim 10^{-6}$), the speedup is more important (*e.g* $\times 5$ with less than 5% actives particles). Moreover, one can also expect a good behavior of the method in Monte Carlo simulations (MC) since the force/energy update via IMCM leads to a $\times 15$ speed-up when 1 or 2 particles are active. This corresponds to a typical scenario of energy computation in a MC trial move.

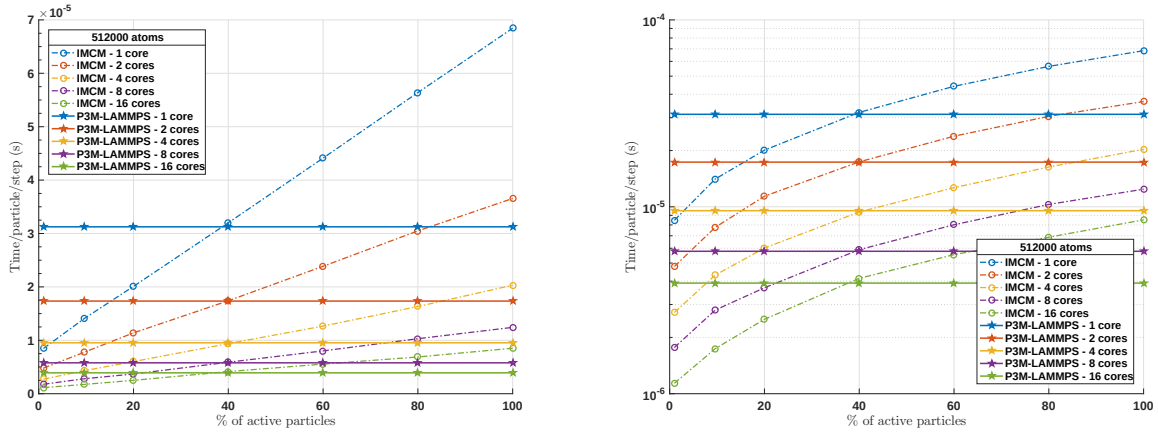


(a) Speedup of IMCM compared to P3M

(b) Speedup of IMCM compared to MCM

Figure 7: System A - Speedup of the Incremental Meshed Continuum compared to Particle Particle Particle Mesh (left) and the Meshed Continuum Method (right).

Parallel Performance



(a) Lin-lin scale

(b) Lin-log scale

Figure 8: Required wall clock time per particle as a function the percentage of active particles for different number of processes. The wall clock time is represented with Linear scale (left) and logarithm scale (right). Performance of LAMMPS P3M is shown as a reference (dotted lines, pentagram marker) — it does not depend on the percentage of active particles. In all cases electrostatics were computed at similar accuracy ($\sim 10^{-5}$).

We assessed the parallel performance of the IMCM on the replicated version of the system A (512000 atoms). IMCM showed a good behavior on a multi-core environment (Figure 8). A 10^{-5} accuracy P3M was used as reference. Linear scale is shown on left In all tests, IMCM becomes faster than P3M when less than about 40% of particles are active. When less than 20% of particles are active, IMCM is 1.5 times faster than P3M on 16 cores. Furthermore, in all configurations, the proposed method is more than twice faster than P3M when the percentage of active particles does not exceed 10%. These results are analogous to the parallel perfomance of ARMD on short-range potentials[39].

The following sections will be dedicated to several practical applications of ARMD where electrostatic interactions were incrementally updated with IMCM.

Sodium chloride with NaCl/ϵ and SPC/ϵ water

The system of sodium chloride in aqueous solution was studied with ARMD. The system was kept at the temperature $T = 298\,\mathrm{K}$ with a Langevin thermostat. Then, we compared several combinations of energy thresholds $(\varepsilon^r, \varepsilon^f)$.

After 10 ns run, the corresponding thermodynamic and dynamic properties of the system were checked against a standard MD simulations at the same order of accuracy. To verify the correctness of our simulations, the hydration of the ions, the structure of the water molecules around ions is computed and checked against results from a standard MD simulations in the same order of accuracy. This hydration can be measured by the four partial radial pair distribution functions (RDFs) g_{Na-H}, g_{Cl-H}, g_{Na-O} and g_{Cl-O}. Although each combination $(\varepsilon^r, \varepsilon^f)$ corresponds to a different average number of active/restrained particles (Table 3) and a distinct temperature profile (Figure 9), the corresponding hydration is similar to results from the reference MD simulation (Figure 10). The peak positions, r_{max}, of the pair distribution functions in our model are given by: $r_{max} \simeq 2.98\,\text{Å}$ for Na-H, $r_{max} \simeq 2.19\,\text{Å}$ and $r_{max} \simeq 3.45\,\text{Å}$ for the first and second peaks of Cl-H, $r_{max} \simeq 2.31\,\text{Å}$ for Na-O and $r_{max} \simeq 3.11\,\text{Å}$ for Cl-O.

The computation of the electrostatics forces was sped up with the use of the IMCM. When

the average percentage of active particles is around 15%, we achieved ×2 speed up relative to a standard MD run. The performance of the method can be increased by restraining not only water particles but also the sodium chloride ions.

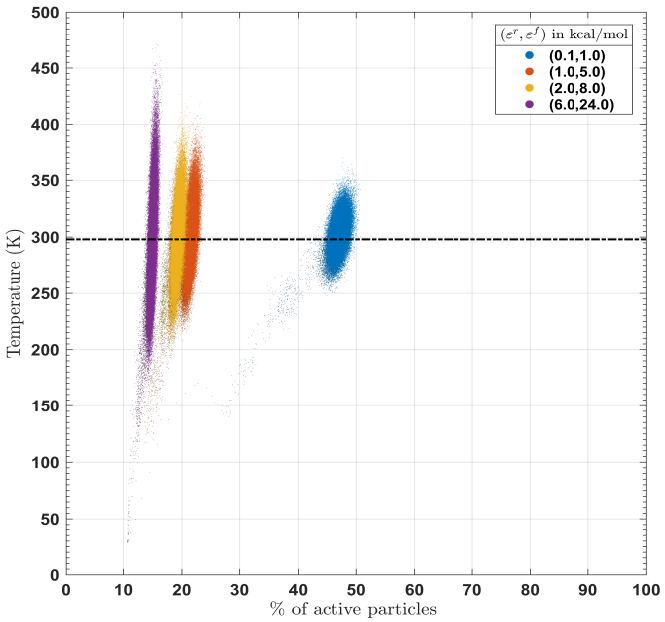
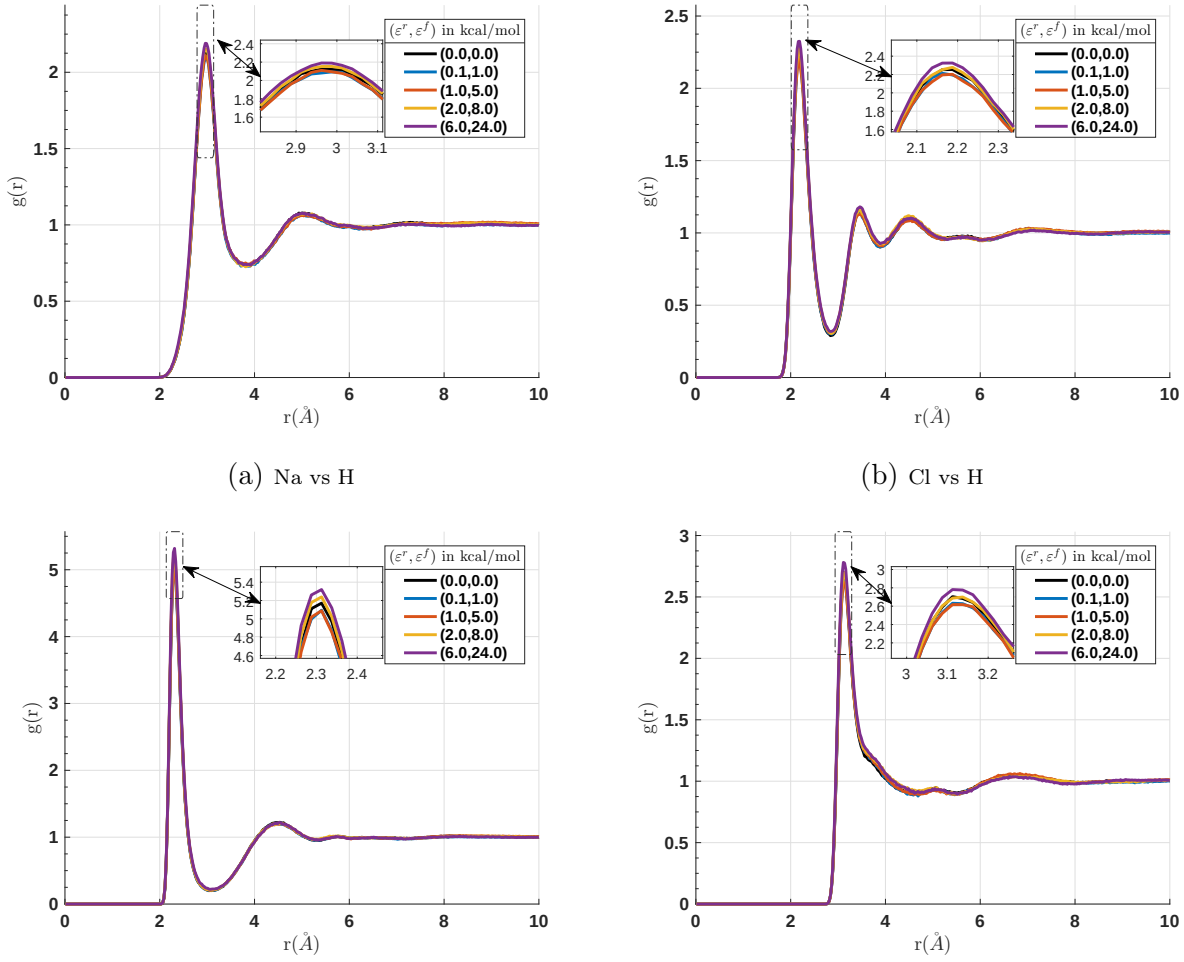


Figure 9: Temperature profile for NaCl + water mixture with an ionic concentration of 10.0 molal. Different restraining parameters $(\varepsilon^r, \varepsilon^f)$ were tested on water molecules. Na and Cl are always active. Black line corresponds to the target temperature (298 K).

$(\varepsilon^r, \varepsilon^f)$ (kcal/mol)	$\langle T \rangle$ (K)	$\langle n_{act} \rangle$ (%)	Speed-up
(0.1,1.0)	298.04	47.02	× 0.90
(1.0,5.0)	298.97	21.56	× 1.54
(2.0,8.0)	297.48	19.37	× 1.63
(6.0,24)	298.73	15.05	× 2.19

Table 3: Speed-up for various combinations of energy thresholds $(\varepsilon^r, \varepsilon^f)$. $\langle n_{act} \rangle$ is the average percentage of active particles. $\langle T \rangle$ corresponds to the average temperature.



(a) Na vs H

(b) Cl vs H

(c) Na vs O

(d) Cl vs O

Figure 10: Ion-water pair distribution functions using ARMD with the NaCl/ϵ force field at 298 K the rigid water model SPC/ϵ and an ionic concentration of 10.0 molal. Different restraining parameters $\left(\varepsilon^r, \varepsilon^f\right)$ were tested on water molecules. Na and Cl are always active. Black line corresponds to a standard molecular dynamics simulation of the system.

Nanopore System

We investigated the ion transport through a graphene nanopore. As aforementioned, a porous graphene monolayer is placed in a 70^3Å^3 box filled with sodium chloride aqueous solution. Chlorine and sodium ions were driven trough the pore with an external electric field perpendicularly oriented to the graphene sheet ($E = 1$ V/Å in minus z-directions).

We designed a 5 Å -diameter functionalized pore by putting charges on its edge. Thus, it behaves like an ionic colander of high selectivity[42]. In our case, the nanopore is terminated by positively charged ions, favoring the passage of chlorine. One may produce this type of pore through ion etching[48].

5 ns simulations were conducted with both ARMD and standard MD. The IMCM was employed in ARMD simulations whereas P3M was used in the standard MD runs. In all cases, the timestep was set to be 2 fs. Once again, the system was maintained at 298 K with a Langevin thermostat. The system C was evolved for 2 ns to achieve equilibrium state. Then, statistics were gathered during the last 3 ns. Cylindrical coordinates (r, z) were employed to analyze the behavior of the nanopore system and the origin was set at the center of the simulation box. The initial configuration corresponds to a quick minimization of a system with uniformly distributed sodium chloride ions and without any external field.

When an external electric field is applied, ions are driven directionally and accumulate on the top ($z > 0$) and bottom ($z < 0$) sides of the graphene layer according to the charge's sign. Sodium ions are driven in the (minus z-direction). With periodic boundary conditions, they appeared on the top of the box. Thus, they accumulated in this region since the pore is impermeable to them. In fact, due to charges located at the edge of the pore, sodium ions are repelled and consequently constrained in the top side of the simulation box (Figure 11). The combination of electrostatic and Lennard-Jones forces leads to the appearance of this concentration polarization layer (CPL). Conversely, the chlorine ions (*moving from bottom side to topside*) are allowed to pass through the pore. However, chlorine ions formed a concentration polarization layer (CPL) adjacent to the graphene sheet since the pore size is relatively small. The same behavior can be observed for sodium ions. It is interesting to observe that chlorine ions seemed to have three preferential ways to enter the pore. This might be related to the geometry of the designed pore. ARMD in addition to IMCM was able to reproduce this behavior (Figure 12), but the average time cost per step is 4 times lesser than standard MD where P3M is used. The ionic distributions of both approaches shows a similar nonuniform ionic distribution with well defined concentration polarization layers (CPLs) identically located for all ionic species. ARMD performs quite satisfactorily

on this system where approximately 93% of particles were adaptively restrained.

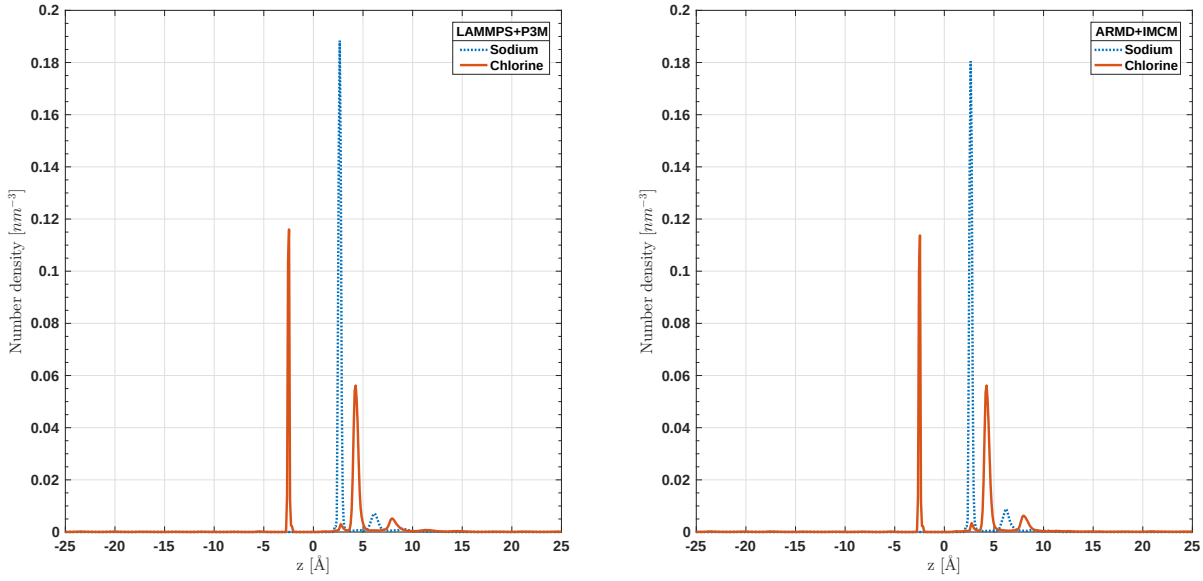


Figure 11: Number density of chlorine (red dotted line) and sodium (blue dashed line) ions along z-axis using standard MD (Left) and ARMD (Right). Both methods show the ion selectivity of the nanopore which is located at $z = 0$.

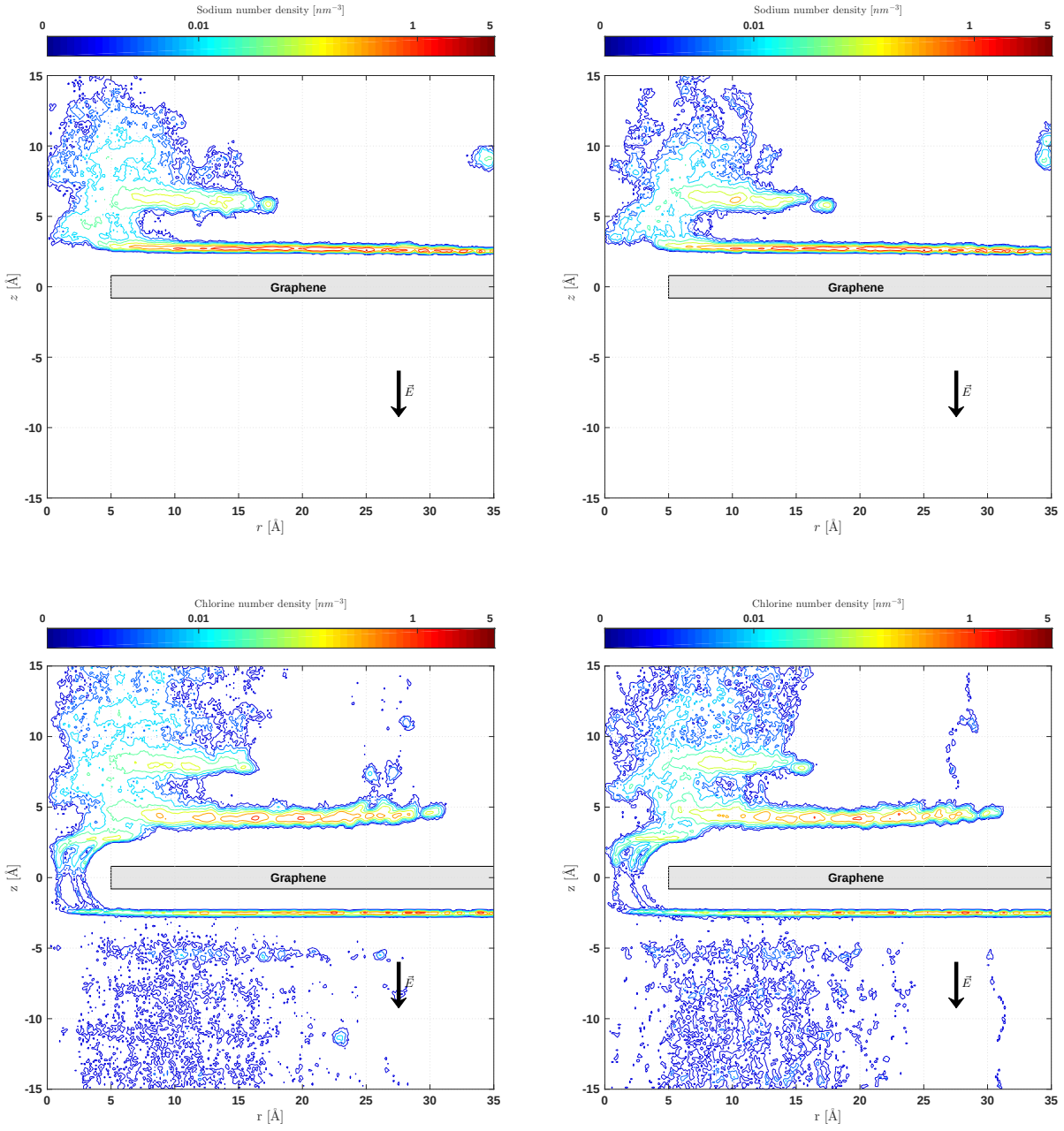


Figure 12: Nonuniform distributions of number density of chlorine (Top) and sodium (Bottom) ions driven by an external electric field (black arrow) $E = 1$ V/Å using standard MD (Left) and ARMD (Right). The gray rectangles at $z = 0$ mark the graphene sheet. Both ions form concentration polarization layers (CPLs).

CONCLUSIONS

In this article, we proposed a novel approach to the computation of electrostatics forces in adaptively restrained particle simulations. This multigrid-based algorithm takes advantage of the fact that ARMD switches positional degrees of freedom on and off during simulations, while letting momenta evolve[20]. We showed that the pairwise electrostatic potential can be incrementally computed. We achieved significant speedups for adaptively restrained simulations, as the number of active particles decreases. We showed that IMCM exhibits a good behavior on both single and multi-core architectures, and we would like to investigate its performance on massively parallel architectures. We expect that the method will scale well in this context thanks to its multigrid foundation.

We also want to examine several theoretical aspects related to the tuning of the method such as the choice of the smooth distribution, the resolution of the mesh and the cutoff radius.

Moreover the proposed method has shown good performance in terms of precision and speed in comparison to the popular P3M. We believe that the IMCM can be combined with ARMD on a wide range of systems. For instance, combining ARMD and IMCM should allow for a more efficient simulation of channeling effects. Our method can also be used in the simulation of a protein in a solvent; where a large share of solvent molecules can be restrained. We thus want to investigate such applications in future works.

ACKNOWLEDGMENTS

We gratefully acknowledge funding from the European Research Council through the ERC Starting Grant n. 307629.

A Construction of smooth charge densities

The meshed continuum method (MCM)[19] requires the use of smooth charge densities. Here, we define a smooth charge density ρ_{r_c} as a normalized radially symmetric distribution such

that $\rho_{r_c}(||\boldsymbol{r}||) = 0$ when $||\boldsymbol{r}|| > r_c$. Some smooth charge densities are described with the help of cardinal B-splines. We denote by $B^{(m)}$ a order-m cardinal B-spline. $B^{(m)}$ is a $\mathcal{C}^{m-2}[0, m]$ function with the following properties[26–28]:

- $supp\left(B^{(m)}\right) = [0, m]$;
- at each interval $[k, k+1]$, $0 \leq k \leq m-1$;
- $\forall t \in [0, m]$, $B^{(m)}(t) = B^{(m)}(m-t)$.

By defining $B^{(m)}_{r_c} : r \mapsto B^{(m)}\left(\frac{m}{2}\left(\frac{r}{r_c}+1\right)\right)$, we can construct a centered B-spline with $supp\left(B^{(m)}_{r_c}\right) = [-r_c, r_c]$. Thereby, a charge density described with a order-m B-spline, reads

$$\rho^{(m)}_{r_c}(\boldsymbol{r}) := \rho_{r_c}(||\boldsymbol{r}||) = \frac{1}{A} B^{(m)}_{r_c}(||\boldsymbol{r}||) \tag{21}$$

where $\frac{1}{A}$ is a normalizing constant. Precisely,

$$\begin{aligned} A &= \int_{\mathbb{R}^3} B^{(m)}_{r_c}(||\boldsymbol{r}||)\mathrm{d}\boldsymbol{r} && (22) \\ &= \int_0^{r_c} 4\pi r^2 B^{(m)}_{r_c}(r)\mathrm{d}r && (23) \end{aligned}$$

The charge density used by Bolten et *al.*[19], in their original paper, is a centered quadratic B-spline. Precisely, this density which is described by a order-3 B-spline, reads,

$$\rho^{(3)}_{r_c}(r) = \begin{cases} \dfrac{-486r^2 + 162r_c^2}{32\pi r_c^5} & \text{if } r < \dfrac{r_c}{3} \\ \dfrac{243(r-r_c)^2}{32\pi r_c^5} & \text{if } \dfrac{r_c}{3} \leq r < r_c \\ 0 & \text{if } r \geq r_c \end{cases} \tag{24}$$

where $r = ||\boldsymbol{r}||$. Using equation (6), the corresponding potential can be determined analytically

$$\phi^{(3)}(r) = \begin{cases} \dfrac{3(81r^4 - 90r^2r_c^2 + 65r_c^4)}{320\pi r_c^5} & \text{if } r < \dfrac{r_c}{3} \\ \dfrac{-243r^5 + 810r^4r_c - 810r^3r_c^2 + 405rr_c^4 - 2r_c^5}{640\pi r r_c^5} & \text{if } \dfrac{r_c}{3} \leq r < r_c \\ \dfrac{1}{4\pi r} & \text{if } r \geq r_c. \end{cases} \tag{25}$$

For this study, we evaluated the use of higher order cardinal splines. For instance, a charge density described by the 5^{th} order B-spline reads

$$\rho_{r_c}^{(5)}(r) = \begin{cases} \dfrac{54(81r^4 - 54r^2r_c^2 + 11r_c^4)}{64\pi r_c^7} & \text{if } r < \dfrac{r_c}{3} \\ \dfrac{-243r^4 + 540r^3r_c - 378r^2r_c^2 + 60rr_c + 17r_c^4}{64\pi r_c^7} & \text{if } \dfrac{r_c}{3} \leq r < \dfrac{2r_c}{3} \\ \dfrac{2187(r - r_c)^4}{64\pi r_c^7} & \text{if } \dfrac{2r_c}{3} \leq r < r_c \\ 0 & \text{if } r \geq r_c, \end{cases} \tag{26}$$

While its corresponding potential reads

$$\phi^{(5)}(r) = \begin{cases} -\dfrac{7290r^6 - 10206r^4r_c^2 + 6930r^2r_c^4 - 3346r_c^6}{4480\pi r_c^7} & \text{if } r < \dfrac{r_c}{3} \\ \dfrac{32805r^7 - 102060r^6r_c + 107163r^5r_c^2 - 28350r^4r_c^3 - 16065r^3r_c^4 + 9933rr_c^6 + 10r_c^7}{13440\pi r r_c^7} & \text{if } \dfrac{r_c}{3} \leq r < \dfrac{2r_c}{3} \\ -\dfrac{3645r^7 - 20412r^6r_c + 45927r^5r_c^2 - 5130r^4r_c^3 + 25515r^3r_c^4 - 5103rr_c^6 + 338r_c^7}{4480\pi r r_c^7} & \text{if } \dfrac{2r_c}{3} \leq r < r_c \\ \dfrac{1}{4\pi r} & \text{if } r > r_c. \end{cases} \tag{27}$$

Alternatively, one can construct densities with the help of high order polynomials. In particular, one can use a restriction of order-$2m$ polynomials defined as follows

$$P_{2m}(r) = \left(r^2 - r_c^2\right)^m \mathbb{1}^{[0,r_c]}(r) \tag{28}$$

where $\mathbb{1}^{[0,r_c]}$ indicates to the characteristic function of $[0, r_c]$. Thus, a smooth charge density described by P_{2m} reads:

$$\rho_{r_c}(||\boldsymbol{r}||) = \frac{1}{A} B_{r_c}^{(m)}(||\boldsymbol{r}||). \tag{29}$$

Again $\dfrac{1}{A}$ defines a normalizing constant such that

$$A = \int_0^{r_c} 4\pi r^2 P_{2m}(r) \mathrm{d}r. \tag{30}$$

For instance, when using an order-10 polynomial, the charge density ρ reads:

$$\rho_{rc}(r) = \frac{9009}{256\pi r_c^{13}} \left(r_c^2 - r^2\right)^5 \qquad \text{if } r \leq r_c. \tag{31}$$

Thus, the corresponding potential reads

$$\phi(r) = \frac{a_{12}r^{12} - a_{10}r^{10}r_c^2 + a_8 r^8 r_c^4 + a_6 r^6 r_c^6 + a_4 r^4 r_c^8 + a_2 r^2 r_c^{10} + a_0 r c^{12}}{8192\pi r_c^{13}} \tag{32}$$

$$\text{where} \qquad a_{12} = 462, \quad a_{10} = 3276, \quad a_8 = 1001,$$
$$a_6 = -1716, \quad a_4 = 18018, \quad a_2 = -12012, \quad a_0 = 6006.$$

Figure 13 shows several densities described by B-Splines (order 3,5,7) or (order 6, 10, 14) polynomials.

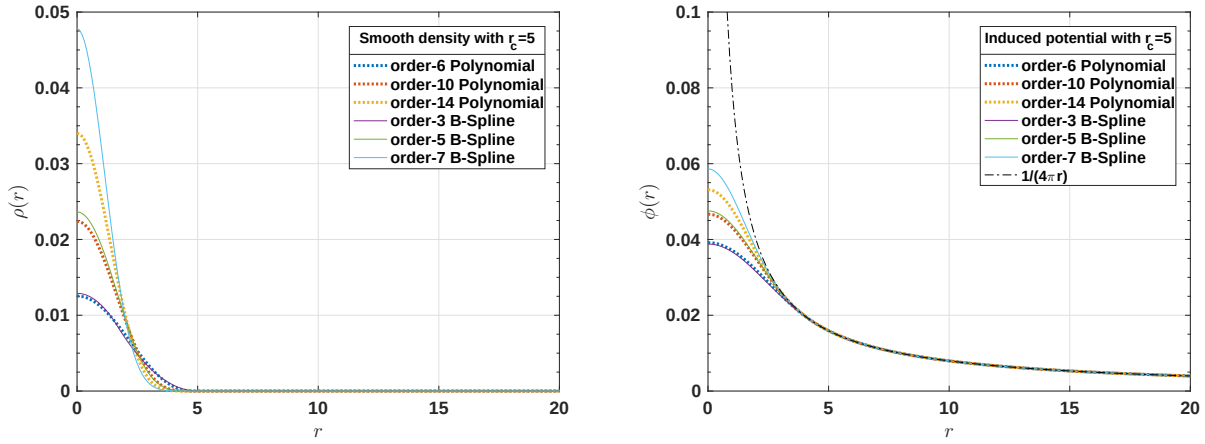


Figure 13: Comparison of various smooth charge densities with $r_c = 5$ (left). Densities described by B-Splines (order 3,5,7) are shown with solid line. Similarly, densities described by order 6, 10 and 14 polynomials are represented with dots. For each configuration, the corresponding potential is plotted (right).

References

1. W. F. van Gunsteren and H. J. Berendsen, Angewandte Chemie International Edition **29**, 992 (1990).

2. M. Karplus and J. A. McCammon, Nature Structural & Molecular Biology **9**, 646 (2002).

3. J. R. Perilla, B. C. Goh, C. K. Cassidy, B. Liu, R. C. Bernardi, T. Rudack, H. Yu, Z. Wu, and K. Schulten, Current Opinion in Structural Biology **31**, 64 (2015), ISSN 1879033X, 15334406, URL http://linkinghub.elsevier.com/retrieve/pii/S0959440X15000342.

4. D. Frenkel and B. Smit, *Understanding Molecular Simulation: from Algorithms to Applications* (Academic Press, 2002), ISBN 9780080519982.

5. A. Arnold, F. Fahrenberger, C. Holm, O. Lenz, M. Bolten, H. Dachsel, R. Halver, I. Kabadshow, F. Gähler, F. Heber, et al., Physical Review E - Statistical, Nonlinear, and Soft Matter Physics **88**, 063308 (2013), ISSN 15393755, URL https://link.aps.org/doi/10.1103/PhysRevE.88.063308.

6. M. Deserno and C. Holm, Journal of Chemical Physics **109**, 7678 (1998), ISSN 00219606, 9807099, URL http://aip.scitation.org/doi/10.1063/1.477414.

7. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, vol. 25 (1988), ISBN 0852743920, URL http://link.aip.org/link/SIREAD/v25/i3/p425/s1{&}Agg=doi.

8. U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, The Journal of Chemical Physics **103**, 8577 (1995), ISSN 0021-9606, URL http://aip.scitation.org/doi/10.1063/1.470117.

9. V. Ballenegger, J. J. Cerdà, and C. Holm, Computer Physics Communications **182**, 1919 (2011).

10. L. Greengard and V. Rokhlin, Journal of computational physics **73**, 325 (1987).

11. C. Sagui and T. Darden, The Journal of Chemical Physics **114**, 6578 (2001).

12. D. S. Cerutti and D. A. Case, Journal of chemical theory and computation **6**, 443 (2009).

13. D. J. Hardy, Z. Wu, J. C. Phillips, J. E. Stone, R. D. Skeel, and K. Schulten, Journal of Chemical Theory and Computation **11**, 766 (2015), ISSN 1549-9618, URL http://pubs.acs.org/doi/abs/10.1021/ct5009075.

14. G. Sutmann, P. Gibbon, and T. Lippert, *Fast Methods for Long-Range Interactions in Complex Systems* (Forschungszentrum Jülich, 2011).

15. S. Aboud, D. Marreiro, M. Saraniti, and R. Eisenberg, Journal of Computational Electronics **3**, 117 (2004).

16. G. Sutmann and B. Steffen, Computer physics communications **169**, 343 (2005).

17. W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, vol. 37 (Society for Industrial and Applied Mathematics, 2000), ISBN 0898714621, arXiv:1011.1669v3, URL http://books.google.de/books?id=oSTGBm64o1AC.

18. S. G. Moore and P. S. Crozier, The Journal of chemical physics **140**, 06B619_1 (2014).

19. M. Bolten, Ph.D. thesis, Wuppertal, Univ., Diss., 2008 (2008).

20. S. Artemova and S. Redon, Physical Review Letters **109**, 190201 (2012), ISSN 00319007, URL https://link.aps.org/doi/10.1103/PhysRevLett.109.190201.

21. M. Bosson, S. Grudinin, X. Bouju, and S. Redon, Journal of Computational Physics **231**, 2581 (2012), ISSN 00219991, URL http://linkinghub.elsevier.com/retrieve/pii/S0021999111007042.

22. Z. Trstanova and S. Redon, Journal of Computational Physics **336**, 412 (2017).

23. K. K. Singh and S. Redon, Modelling and Simulation in Materials Science and Engineering (2017), ISSN 1361651X, URL http://iopscience.iop.org/article/10.1088/1361-651X/aa6e36{%}5Cnhttp://iopscience.iop

24. T. Schlick, *Molecular Modeling and Simulation: An Interdisciplinary Guide*, vol. 21 (Springer-Verlag New York, Inc., 2010), ISBN 978-1-4419-6350-5, 1406.6401.

25. M. Griebel, S. Knapek, and G. Zumbusch, *Numerical simulation in molecular dynamics. numerics, algorithms, parallelization, applications, volume 5 of texts in computational science and engineering* (2007).

26. C. K. Chui, *An introduction to wavelets* (Elsevier, 2016).

27. C. K. Chui, *Wavelets: a mathematical tool for signal analysis*, vol. 1 (Siam, 1997).

28. G. V. Milovanović and Z. Udovičić, Applied Mathematics Letters **23**, 1346 (2010), ISSN 08939659, URL http://dblp.uni-trier.de/db/journals/appml/appml23.html{#}MilovanovicU10.

29. U. Ananthakrishnaiah, R. Manohar, and J. W. Stephenson, Numerical Methods for Partial Differential Equations **3**, 229 (1987), ISSN 0749-159X, URL http://doi.wiley.com/10.1002/num.1690030307.

30. U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid* (Academic Press, Inc., 2000), ISBN 012701070X.

31. A. Brandt and O. E. Livne, *Multigrid Techniques : 1984 Guide With Applications to Fluid Dynamics*, Classics in Applied Mathematics (Society for Industrial and Applied Mathematics, 2008), ISBN 1611970741, URL https://books.google.fr/books?id=EvClMX03MyAC.

32. W. Gautschi, *Numerical analysis* (Springer Science & Business Media, 2011).

33. H. R. Schwarz and J. Waldvogel, *Numerical analysis: a comprehensive introduction* (John Wiley & Sons, 1989).

34. K. K. Singh and S. Redon, Journal of Computational Chemistry (2017), ISSN 1096-987X, URL http://dx.doi.org/10.1002/jcc.25126.

35. S. Plimpton, P. Crozier, and A. Thompson, Sandia National Laboratories **18** (2007).

36. S. Plimpton, Journal of Computational Physics **117**, 1 (1995), ISSN 00219991, URL http://linkinghub.elsevier.com/retrieve/doi/10.1006/jcph.1995.1039.

37. M. Hülsemann, Frankand Kowarschik, M. Mohr, and U. Rüde, *Parallel Geometric Multigrid* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 165–208, ISBN 978-3-540-31619-0, URL https://doi.org/10.1007/3-540-31619-1_5.

38. E. Chow and R. Hu, Parallel processing for scientific computing **20**, 179 (2006).

39. K. K. Singh, D. F. Marin, and S. Redon, in *High Performance Computing & Simulation (HPCS), 2017 International Conference on* (IEEE, 2017), pp. 308–314.

40. G. Hu, M. Mao, and S. Ghosal, Nanotechnology **23**, 395501 (2012).

41. M. E. Suk and N. Aluru, The Journal of Physical Chemistry Letters **1**, 1590 (2010), http://dx.doi.org/10.1021/jz100240r@proofing, URL http://dx.doi.org/10.1021/jz100240r@proofing.

42. K. Sint, B. Wang, and P. Král, Journal of the American Chemical Society **130**, 16448 (2008).

43. Q. Ying-Hua, L. Kun, C. Wei-Yu, S. Wei, T. Qi-Yan, and C. Yun-Fei, Chinese Physics B **24**, 108201 (2015).

44. R. Fuentes-Azcatl and M. C. Barbosa, The Journal of Physical Chemistry B **120**, 2460 (2016).

45. S. Plimpton, R. Pollock, and M. J. Stevens, in *Proceedings of the Eighth Siam Conference on Parallel Processing for Scientific Computing* (Citeseer, 1997), pp. 1–13.

46. B. Fornberg, Mathematics of computation **51**, 699 (1988).

47. R. D. Skeel, D. J. Hardy, and J. C. Phillips, Journal of computational physics **225**, 1 (2007).

48. F. Molitor, J. Güttinger, C. Stampfer, D. Graf, T. Ihn, and K. Ensslin, Phys. Rev. B **76**, 245426 (2007), URL https://link.aps.org/doi/10.1103/PhysRevB.76.245426.